# BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

## IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.**

The Patent Office
Concept House
Cardiff Road
Newport
South Wales
NP10 8QQ



I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with the patent application identified therein.

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.

Signed K A R Clements

Dated 4 February 2002

Patents Form 1/77

Patents Act 1977
(Rule 16)

The Patent Office

THE PATENT OFFICE
E
12 APR 2001
LONDON

& CO

# Request for a grant of a patent

*(See the notes on the back of this form you can also get an explanatory leaflet from the Patent Office to help you fill in this form)*

**The Patent Office**

Cardiff Road
Newport
Gwent NP9 1RH

1. Your reference — P011319GB

2. Patent application number *(The Patent Office will fill in this part)* — **0109283.2**

12 APR 2001
17APR01 E621991-9 D02246
P01/7700 0.00-0109283.2

3. Full name, address and postcode of the or of each applicant *(underline all surnames)*

ARM Limited
110 Fulbourn Road
Cherry Hinton
Cambridge
CB1 9NJ
United Kingdom

Patents ADP number *(if you know it)* 7498124002

If the applicant is a corporate body, give the country/state of its incorporation — United Kingdom

4. Title of the invention — TESTING FOR COMPLIANCE

5. Name of your agent *(if you have one)* — **D YOUNG & CO**

"Address for service" in the United Kingdom to which all correspondence should be sent (including the postcode) — **21 NEW FETTER LANE**
**LONDON**
**EC4A 1DA**

Patents ADP number *(if you know it)* — **59006**

6. If you are declaring priority from one or more earlier patent applications, give the country and date of filing of the or each of these earlier applications and *(if you know it)* the or each application number

| | Country | Priority application number *(if you know it)* | Date of filing *(day/month/year)* |
|---|---|---|---|
| **1st** | | | |
| **2nd** | | | |
| **3rd** | | | |

7. If this application is divided or otherwise derived from an earlier UK application, give the number and filing date of the earlier application

| Number of earlier application | Date of filing (day/month/year) |
|---|---|
| | |

8. Is a statement of inventorship and of right to grant of a patent required in support of this request? *(Answer 'Yes' if:*
*a) any applicant named in part 3 is not an inventor, or*
*b) there is an inventor who is not named as an applicant, or*
*c) any named applicant is a corporate body.*
*See note (d))*

Yes

9. Enter the number of sheets for any of the following items you are filing with this form. Do not count copies of the same document

| | |
|---|---|
| Continuation sheets of this form | None |
| Description | 5 |
| Claim(s) | 0 |
| Abstract | 0 |
| Drawing(s) | 0 |

10. If you are also filing any of the following, state how many against each item

| | |
|---|---|
| Priority Documents | 0 |
| Translation of Priority Documents | 0 |
| Statement of inventorship and right to grant of a patent *(Patents Form 7/77)* | 0 |
| Request for preliminary examination and search *(Patents Form 9/77)* | 0 |
| Request for substantive examination *(Patents Form 10/77)* | 0 |
| Any other documents *(Please specify)* | 0 |

11. I/We request the grant of a Patent on the basis of this application.

Signature D Young & Co — Date

**D YOUNG & CO**
Agents for the Applicants

12 Apr 2001

12. Name and daytime telephone number of person to contact in the United Kingdom — David Horner — 023 80719500

**Warning**

*After an application for a patent has been filed, the Comptroller of the Patent Office will consider whether publication or communication of the invention should be prohibited or restricted under Section 22 of the Patents Act 1977. You will be informed if it is necessary to prohibit or restrict your invention in this way. Furthermore, if you live in the United Kingdom, Section 23 of the Patents Act 1977 stops you from applying for a patent abroad without first getting written permission from the Patent Office unless an application has been filed at least 6 weeks beforehand in the United Kingdom for a patent for the same invention and either no direction prohibiting publication or communication has been given, or any such direction has been revoked.*

**Notes**

*a) If you need help to fill in this form or you have any questions, please contact the Patent Office on 01645 500505.*

*b) Write your answers in capital letters using black ink or you may type them.*

*c) if there is not enough space for all the relevant details on any part of this form, please continue on a separate sheet of paper and write "see continuation sheet" in the relevant part(s) Any continuation sheets should be attached to this form.*

*d) If you answered 'Yes' Patents Form 7/77 will need to be filed.*

*e) Once you have filled in the form you must remember to sign and date it.*

*f) For details of the fee and ways to pay please contact the Patent Office.*

# Testing for Compliance

A.Nightingale. *Verification Manager, ARM IP Solutions Division*

*Abstract*—**AMBA Compliance enables the developer of an IP component to demonstrate that the testing of the AMBA interface has achieved a pre-defined quality level.**

*Index Terms*—**ARM, AMBA, Advanced Micro-controller Bus Architecture, AHB, AMBA High-speed Bus, APB, AMBA Peripheral Bus.**

## I. Introduction

THE AMBA specification is an on-chip bus specification which is intended to allow the rapid building of System-on-Chip (SoC) devices by the integration of a number of different IP components. The AMBA specification enables this process by providing a standard interface, which allows the IP supplier to design and test the module without any knowledge of the system in to which the component will be finally integrated.

In a given company/design house, IP components that are used to construct a SoC may be obtained from a number of different suppliers, including: i) *Third party IP suppliers such as ARM*, ii) *other internal design departments*, iii) *the SoC design team itself.*

AMBA Compliance enables the supplier of an IP component to demonstrate that the testing of the interface has achieved a pre-defined quality level and this, in turn, gives the SoC integrator confidence that the component will function correctly when integrated in to the rest of the SoC design.

## II. Overview Of Compliance Testing

AMBA Compliance Testing is performed using a simulation environment in which the IP component is exercised. The simulation environment also contains the following two elements:

### A. Protocol Checker

Cycle by cycle checks to ensure that the device under test obeys the rules within the AMBA specification. The protocol checks are a fixed set of rules from the AMBA specification and a component must not violate any rules in order to claim AMBA compliance. There is no configuration of the protocol rules. An example of an AMBA AHB Master protocol rule is, "*In the second cycle of a Split or Retry response the master must drive **HTRANS** to IDLE*".

### B. Coverage Monitor

The coverage points are a list of various bus transactions that must be observed within the test environment before compliance can be claimed. The aim of coverage points is to ensure that the tests have sufficiently exercised the component under test before compliance can be claimed. An example of a coverage point is, "*The bus slave must be accessed by a read transfer followed immediately by a write transfer to the same address*".

The advantage of using coverage points, compared to a standard pre-defined sequence of bus transactions, is that it allows for the fact that a particular IP component may require a specific sequence of transactions– in order to fully exercise the bus interface.

## III. Compliance Test Environment

Two typical environments are shown in the diagrams below. In the first environment *Fig. 1* the device under test is connected to a standard AMBA testbench, which reads the stimulus generation and response checking information from a User Defined Test Sequence file.

It is important to note that the Protocol Checker and the Coverage Monitor **do not** automatically check the integrity of the data transferred during compliance tests. The user should ensure that the device under test (DUT) *operates* correctly during a 'compliance run'. When configured as in Fig.1, the compliance test environment aids in the operational testing of the DUT by supporting 'data checking commands' in the User Defined Test Sequence input file.

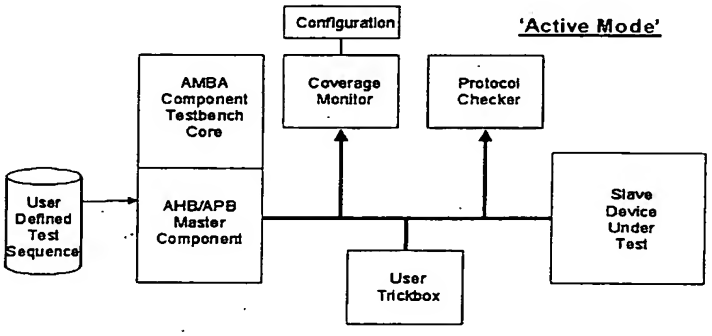


Fig. 1. †On completion of *test*, reports test cases that have not been observed. ‡ During *test*, reports protocol violations. In this environment, the user supplies: '*Configuration*', '*User Defined Test Sequence*' and an RTL implementation of '*Device Under Test*'.

In the second case, a *system simulation environment* is used to provide the stimulus and response checking for the device under test. This environment still retains the *Coverage Monitor* and the *Protocol Checker*. This is illustrated in *Fig. 2* below:

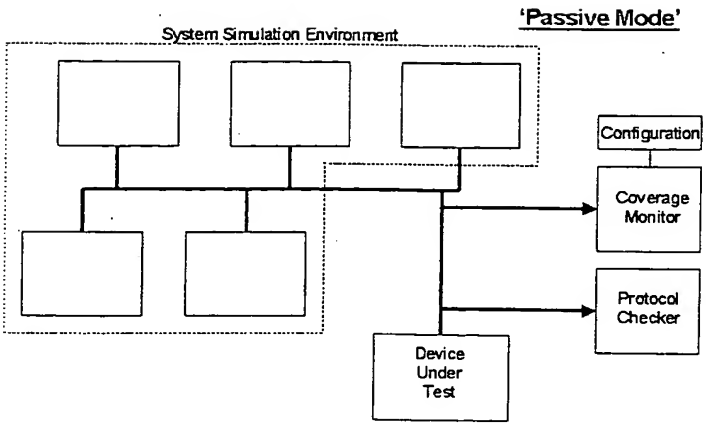


Fig. 2. Inside dotted boundary is a *System Simulation Environment*. In this *Passive Mode* environment, the user supplies: *'Configuration'*, *'System Simulation Environment'* and an RTL implementation of *'Device Under Test'*.

## IV. Claiming AMBA Compliance

AMBA Compliance is a self-certification process. The IP supplier is responsible for performing the tests for compliance and making a suitable declaration, as described below:

**The Declaration of AMBA Compliance must include:** *a) AMBA Specification revision number. b) Compliance configuration. c) Explanation of any compliance coverage points not tested.*

ARM has developed an AMBA Compliance Toolkit (ACT) for the sole purpose of enabling an IP supplier to generate and ship an AMBA IP compliance solution.

An IP Supplier who claims AMBA compliance for a particular IP component is able, using the ACT toolkit, to supply a testbench environment including any required vectors, enabling the User to repeat the Compliance test. Any User who has access to the IP component may request this.

*Note however that the ACT supplied to repeat the compliance test, is locked per IP component, and* ***cannot*** *be configured therefore to test generic IP.*

## V. Compliance Test Configuration

There is no configuration of the Protocol Rules, and an IP component **must not violate** any of the AMBA Protocol Rules in order to claim compliance.

However, the *Coverage Points* may be configured to allow for the fact that certain AMBA modules will not utilise all the features of the bus. It is important to note that the allowable configurations for a module are defined so that all modules will always work together correctly.

As a basic rule *'an AMBA component must be able to accept all possible input combinations, but it does not have to generate all possible output combinations'*.

To illustrate this, a **bus master** that has input signals for *Transfer Ready*, *Transfer Response* and *Bus Grant* must be able to deal with all possible combinations of these inputs. However, it is not required to generate all possible output combinations, so it is only required to generate the types of burst and transfer sizes that are appropriate to its operation.

It should be noted that turning off certain coverage points would enable *additional protocol checks*. For example, if a bus master compliance check is configured for a bus master which does not support 4-beat wrapping bursts this configuration option will have the effect of removing the coverage points to observe various types of 4-beat wrapping burst. It will however enable an additional check to ensure that the master never performs this type of transfer.

A **slave** must also be capable of accepting all possible inputs and must deal correctly with all the different combinations of transfers. However, it is acceptable to build slaves, which do not utilise all the transfer information that is available on the bus. A typical example of this would be a slave, which does not use the burst information. To simplify the compliance testing of such slaves, if the slave does not have certain signals as inputs then the coverage points related to the different combinations of that input could be omitted.

## VI. ACT Features

In summary so far then, we have established that a given ACT configuration can operate either as in Fig. 1 or in a System Simulation Environment as illustrated in Fig. 2 (the Protocol Checker and the Coverage Monitor components being common to both scenarios). The above two ACT operating modes are termed as **Active mode**, and **Passive mode** respectively.

In order to specify these configurations, ARM has also implemented a powerful yet simple ***'Configuration API'***, which not only describes what signal names a DUT has, but also the capabilities of the DUT itself for coverage configuration. A further feature of the Configuration API is to enable access to generic parameters such as: input and output file specifications, bus-width, endianess, debug and error reporting. Only one Configuration API input file is required for multiple ACT configurations; this is especially useful for Passive Mode; although multiple configurations are feasible for Active Mode, only single DUT example scenarios are currently provided with the ACT package.

### A. Active Mode Features

An Active Mode ACT configuration greatly reduces the burden of having to generate a point-solution testbench for each user DUT; in fact the ACT itself generates all the necessary AMBA *'fabric'* around the DUT in order to allow user supplied stimulus to be driven onto the DUT during the process of compliance testing.

RTL ***'trickbox'*** examples are provided for interfacing with the DUT. The trickbox allows non-AMBA signals of the DUT to be controlled and observed. The trickbox has a 32-bit output port and a 32-bit input port, as well as standard AHB/APB slave interface to control the trickbox. The AHB/APB interface means that the trickbox can be instantiated in the testbench and is controlled by using test vectors, which access the trickbox, rather than the DUT.

Active mode configurations will support the following devices: **AHB Master, AHB Slave, APB Master (Bridge), APB Slave, Arbiter and Decoder.**

In order to drive Masters or supply Slave responses, Active Mode also provides the facility for reading stimulus from vector files specified in the Configuration API. The format of these stimulus files has been specifically designed such that they can either be hand-written, or more conveniently auto-generated using languages such as Perl or C.

Depending on the configuration/type of DUT, certain additional facilities may also be enabled from the ACT. In the case of the AHB Master configuration, a behavioural memory-slave device is generated that has both memory initialisation file, and itself generates a memory-dump file of the same format. This configuration can be observed in Fig. 3 below:

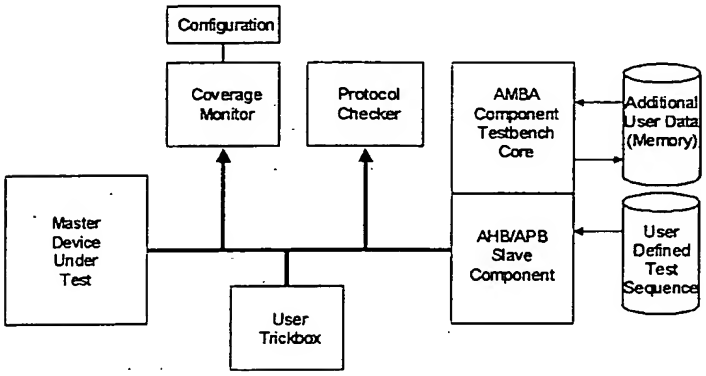


Fig. 3. The master DUT accesses the bus as if it were communicating to a slave device. However the environment created around it drives back directed responses. The *trickbox* component is muxed in with the slave response signals to enable additional behaviour.

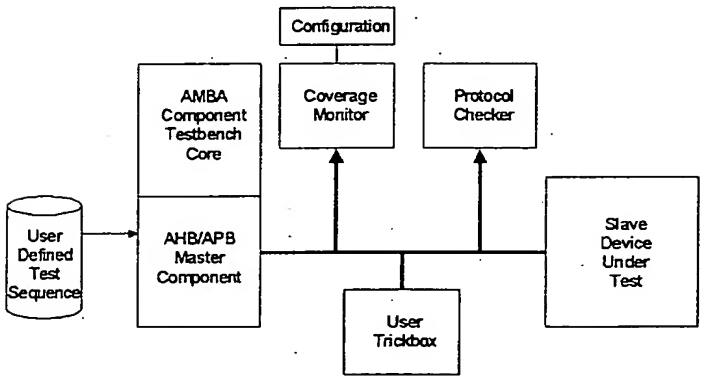


Fig. 4. The slave DUT accesses the bus as if it were responding to a master device. However the environment created around it drives directed responses. The *trickbox* component is also muxed in with the slave response signals to enable additional behaviour. For example in AHB mode, the *trickbox* may provide waited responses as another slave device.

### B. Passive Mode Features

Aimed primarily at an in-system test environment, passive mode effectively disables the concept of ACT 'auto-generated AMBA fabric', and stimulus-driving capabilities, as is otherwise available in Active Mode. As mentioned above, the same Configuration API file is used for specifying details about the DUT itself. In this mode however, any Active Mode specific configuration options are ignored.

Passive mode is also ideal for verifying legacy designs, as the user does not need to isolate the DUT into an Active Mode testbench for compliance testing. It must be noted however that achieving full coverage is possibly more challenging in Passive Mode, if the source of stimulus to the DUT has a limited transaction or scenario generating capability. A more likely scenario would be to ensure that the DUT did not generate any AMBA protocol violations over the course of a sequence of test runs.

In either mode of operation, a coverage report is generated for optional viewing at the end of a run. This is in the form of a **GUI** that displays both the *completed* and *outstanding* compliance coverage goals associated with the *DUT* and its particular capabilities configuration. It is from this data that an *'AMBA Compliance Certificate'* can be generated.

## VII. Configuration API Detail

The ability to specify multiple ACT configurations from a single configuration file is very useful. Conventional methods rely on a deep knowledge of the signal mappings and capabilities of the DUT, which ultimately lead to 'point solution' scenarios and hence reduced testbench re-usability. Even if a specific toolkit is developed for test, there is an implicit degree of re-configuration and 'tweaking' to get a particular DUT to sit comfortably within the test environment. ARM has addressed this issue using a three-step solution.

### A. Thin RTL DUT interface module – Active Mode Only

When the DUT is submitted for testing in ACT Active Mode, it is declared and instantiated in a simple RTL top-level module. This module 'interfaces' to a full ACT AMBA test environment, which is dynamically created at simulation time (see part *C.* below), depending on the ACT configuration file supplied. The interface module also provides optional ***clock*** and ***reset*** signals, along with an optional trickbox interface and associated signal muxing between RTL and the ACT AMBA test environment itself.

### B. User Configuration file

For each ACT operational mode (one of: **AHB_MASTER, AHB_SLAVE, APB_MASTER, APB_SLAVE, ARBITER, DECODER**, either in Active Mode or Passive Mode), example User Configuration files are provided.

The remaining entries in each section provide the following configuration information to the ACT: i) *DUT Name and optional ID,* ii) *HDL Path/Level of hierarchy of the DUT in the design* iii) *DUT signal map e.g. HCLK=HCLKsys, or HCLK=~/top/HCLK to override* ii) *above.* iii) *General testbench parameters e.g. ENDIANESS=LITTLE, ERRORSTOP=YES, VERBOSITY=ERRORS_ONLY.* iv) *Coverage configuration e.g. DIRECTION=NO_WRITE for devices that are read-only.*

### C. Configuration API Engine

As part of the ACT initialization sequence, the User

Configuration file is read in and parsed. If no configuration errors are found (e.g. missing signal definitions, invalid input files, bad parameters), the Configuration API Engine dynamically instantiates the required ACT test components. For example in AHB Master Active Mode, **Arbiter**, **Decoder**, **Default Master**, and **Slave** models are generated and automatically interfaced to the DUT. In AHB Slave Active Mode, a **Decoder** and a **Master** model are generated.

## VIII. Protocol Checking Detail

The basic principle of protocol checking an AMBA bus is to employ a sequence of checks for ensuring that the AMBA bus protocol is observed during transactions, which comprise of a number of bursts, of individual transfers. The current implementation of ACT uses temporal expressions to allow simulation events to trigger heuristics that describe these protocol checks. Another approach that could be adopted is that of Property Checking, however this particular methodology is still currently under evaluation. An extract from ARMs property checking tool evaluation programme is given in part ***B.*** of this section.

### *A. Temporal Expressions*

A temporal expression is a combination of events and temporal operators that describes behavior. A temporal expression captures temporal relationships between events, values of fields, variables, or other items during a test.

Due to the large and varying number of temporal expressions used in the ACT, only a brief concept view of how these constructs are used is described here.

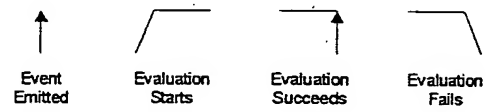


Fig. 5. Legend for Temporals Graphics

The *Verisity Specman™ 'e'* language was chosen amongst other reasons for its rich temporal language implementation. As an example, the following AMBA AHB sequence evaluation illustrates the monitoring of *hgrant* events with respect to the ***HTRANS*** signal being set to *IDLE* (an *htrans.idle* event).

With reference to Fig.6, the *hclk* event is the sampling event for the two sequences involving the *hgrant* and *htrans.idle* events. The *hclk* event itself is emitted at the rise of the ***HCLK*** signal.

The {@hgrant;@htrans.idle} sequence starts evaluating each time *hgrant* occurs at *hclk*. When *htrans.idle* is sampled one *hclk* after *hgrant*, the sequence succeeds.

The {@hgrant;[1];@htrans.idle} sequence also starts evaluating at the first *hgrant* occurrence at hclk, and shifts at the next occurrence of *hclk* due to the *'followed by 1 cycle'* clause ([1]). On the third occurrence of *hclk*, the sequence succeeds. Note however that the temporal expression does not evaluate after the 5th cycle, because of the absence of the *htrans.idle* event.

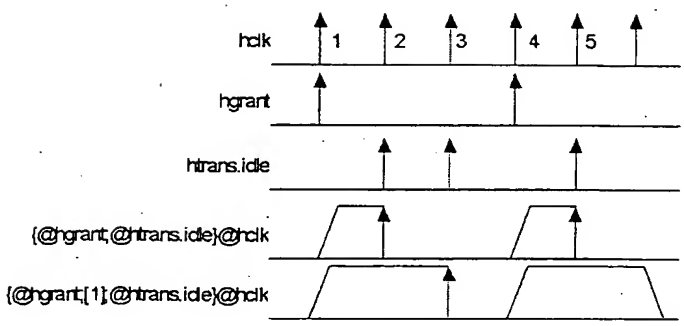


Fig. 6. Example graphical notation to represent evaluation of temporal sequences. Note that some events relate directly to signal values, other events are defined from a combination of signal states sampled at the reference event – the rising edge of ***HCLK*** in the case of AMBA AHB.

In this way, protocol errors can typically be defined by using events such as those described above to capture a condition where a particular rule is not being observed. These events can be used to either build further event sequences, or they can be logically combined to generate assertion statements. A typical form for an ACT protocol rule may be:

**expect <*rule*> is {<*temporal-expression*>}@hclk;**

### *B. Property checking techniques*

ARM is currently evaluating Property Checkers, EDA tools based on a formal mathematical techniques. Formal property checking involves writing a property that closely matches a system requirement, and then proving *exhaustively* that the property holds on the design. The exhaustive nature of this technique means that it can find bugs that are not covered by functional simulations.

A particular property checking EDA tool (that is also currently under evaluation at ARM), was used to implement the AMBA AHB protocol rules, which were written as formal properties and proven on an ARM core *Bus Interface Unit* (BIU), which was in turn connected to an AMBA bus. Property checkers work best at the block levels, with blocks under 100k gates being ideal (the BIU is less than 20k gates).

The property language for the tool being used has a *Verilog* syntax with just a few temporal constructs that allow you to set or check signals at particular clock cycles (relative to some starting clock cycle). An example of such a property is a requirement for the ***HTRANS*** signal output to be *IDLE* or *NON-SEQUENTIAL* if the AHB master is not granted. The property was written to match the following timing diagram:

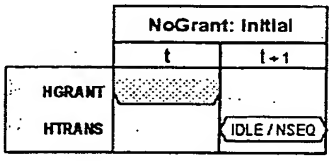


Fig. 7. Timing diagram to represent the HGRANT/HTRANS property.

Times **t** and **t+1** in the timing diagram in Fig. 7 represent clock cycles. The dotted value is an *assumption* and the following *IDLE/NSEQ* part is a proof *obligation*. The timing diagram is saying that *if* the ***HGRANT*** signal is low in one clock cycle (time t), *then* the ***HTRANS*** signal must be *IDLE* or *NSEQ* in the following clock cycle. The proof is exhaustive, so time **t** can be any *arbitrary* clock cycle. You can think of property checking in terms of a simulation metaphor, imagine sliding the timing diagram over an exhaustive simulation – whenever the assumptions hold the obligations *must* hold.

If your property passes then your RTL will always satisfy this requirement. If the property fails, then the property checker outputs a short '*VCD waveform*' file for debugging.

The debug waveform in this case highlighted a fail where the ***HGRANT*** signal was low at time **t** but the ***HTRANS*** signal was *SEQ* at **t+1**. The property failing was the correct result but there was no bug in the RTL – the property itself was incomplete. It turned out after further examination of the AMBA AHB Specification Rev.2, that if the AHB Master is waiting (***HREADY*** signal input *low*) then ***HTRANS*** should be maintained. The property was corrected to match the following timing diagram:

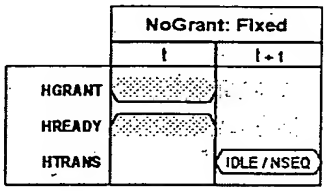


Fig. 8. Timing diagram to qualify the HGRANT/HTRANS property with the **HREADY** signal.

Note that this property says that *if* the master is not granted and is not waiting, *then* the ***HTRANS*** signal must be *IDLE* or *NSEQ* in the following clock cycle. The corrected property was proven, in just a few seconds of CPU processing time.

On a final note, one of the interesting challenges of property checking is to develop the ability to easily identify valid starting states. In one evaluation, it was noted that a particular protocol violation was caused only because one of the exhaustive starting conditions was identified as '*impossible to reach*'.

## IX. Coverage Monitor Detail

Coverage is the process of observing specific sequences of AMBA bus activity. Again, the *Verisity Specman*™ tool enables the ACT to achieve this through the definition of '*coverage groups*'. A coverage group is a program structure/object member that contains a list of data items for which data is collected over time.

A useful feature of the *Specman*™ tool is the ability to perform cross-coverage; this allows observation of interaction *between* coverage data items. An AMBA AHB Master example of cross-coverage usage would be to ensure that for each burst type (*SINGLE, INCR, WRAP4, INCR4, WRAP8, INCR8, WRAP16, INCR16*), has different slave response types observed for both read and write bursts.

From the User Coverage Configuration, the ACT coverage engine decides on what features to cover, and what protocol rule(s) need to be added/ignored during testing.

At the end of a test execution run, all coverage information is collated into a GUI that enables the user to identify how much of the required coverage space a DUT has explored.

Coverage goals are plotted as Red/Amber/Green histograms, and should sufficient coverage goals be reached, a certificate is produced that qualifies the DUT as ***AMBA compliant***. Where the DUT configuration specifies that certain features are not supported, these are noted as exceptions on the compliance certificate. Fig. 9 shows a typical coverage GUI for an AHB Slave DUT.

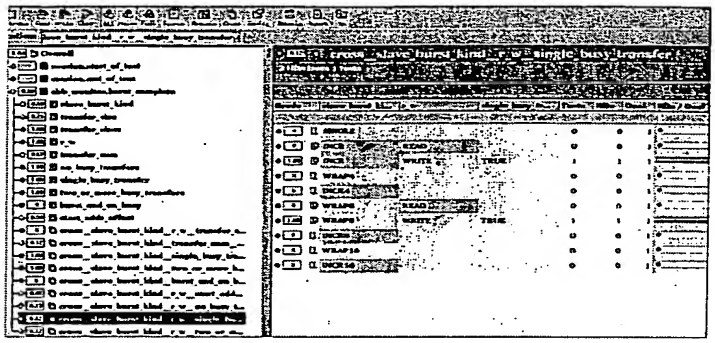

Fig. 9. Example coverage GUI output for an AHB Slave DUT. Each aspect of coverage can be viewed, along with coverage '*holes*' that have not yet been observed using the current test environment/stimulus file.

## References


[1] ARM, "AMBA Specification (Rev 2.0)" *ARM Document No. ARM IHI 0011A*, http://www.arm.com May. 1999.

[2] J. Bergeron, "Writing Testbenches, *FUNCTIONAL VERIFICATION OF HDL MODELS" Kluwer Academic Publishers.* Boston/Dordrecht/London, ISBN 0-7923-7766-4, 2000.

[3] L. Benning, and H. Foster, "Principles of Verifiable RTL Design. *A FunctionalCoding Style Supporting Verification Processes in Verilog" Kluwer Academic Publishers.* Boston/Dordrecht/London, ISBN 0-7923-7788-5, 2000.

[4] M. Stuart, and D. Dempster, *Foreword by Ellis C.Smith* "Verification Methodology Manual, *For Code Coverage In HDL Designs"Teamwork International.* United Kingdom, ISBN 0-9538-4820-5, August 2000.

[5] Verification Development Working Group, "VSI Alliance, Taxonomy of Functional Verification For Virtual Component Development and Integration (TFV 1.0)", Member Review Revision 1, http://www.vsi.org/library/specs/summary.htm 25 Sep 2000.